Report No. 316

ILLIAC IV

QUARTERLY PROGRESS REPORT

October, November and December, 1968

Contract No.
US AF 30(602)4144

ILLIAC IV Doc. No. 216

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

ILLIAC IV

QUARTERLY PROGRESS REPORT

October, November and December, 1968

Contract No.
US AF 30(602)4144

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois
61801

February 1, 1969

This work was supported in part by the Department of Computer Science, University of Illinois, Urbana, Illinois, and in part by the Advanced Research Projects Agency as administered by the Rome Air Development Center, under Contract No. US AF 30(602)4144.

TABLE OF CONTENTS

REPORT SUMMARY

The following major problems became apparent during this quarter.

Semiconductor Devices

As a result of detailed investigation and continuing pressure by Burroughs, information was obtained that Texas Instruments (TI) was in considerable difficulty developing the MSI devices for ILLIAC IV. The problems TI encountered fell into the categories of testing equipment and obtaining multilayer substraits. The ILLIAC IV MSI devices are 80 pin devices and the only available tester is a 44 pin tester. TI attempted to purchase other testers but the subcontractors were not able to produce. TI had contracted with American Lava to supply ceramic substraits. For a variety of reasons American Lava was not able to produce. As a result of the above problems, the PE contract with TI was terminated and an investigation began on replacing the MSI devices. The only solid technology base available to the ILLIAC IV Project were DIL's (Dual In Line). The decision was made to use the DIL's being produced by TI as part of the ILLIAC IV contract. This decision impacts the processing element as the PE was being designed with both MSI devices and DIL's. The Control Unit (CU) had not been designed with MSI devices. Burroughs is in the process of developing a new mechanical configuration for the processing element and making the appropriate modifications to the design to incorporate the details.

Memories Systems

A review of Burroughs' plans and projected cost in producing the thin film memories created a situation whereby the University felt obligated to require Burroughs to solicit bids from industry to provide backup for the thin film memories program. Burroughs issued a

request for quotation to memory suppliers, including all the major semiconductor manufacturers. The responses were from semiconductor manufacturers and not a single thin film memory system was bid. In addition, the firm fixed price bids were approximately one-third (1/3) the cost of the proposed thin film system. As a result, the thin film memory program has been terminated at Burroughs and a procurement activity started for semiconductor memories. At this time, Motorola, TI, and Fairchild are providing quotations on ILLIAC IV memories.

Impact on Cost, Schedule and Performance

The retrenchment to a DIL technology will have impact on cost. This impact is in the process of being estimated by Burroughs. Schedule impact is estimated to be in the neighborhood of six (6) months which will provide the first quadrant of ILLIAC IV delivered to the University in June of 1970. The implementation with DIL's will result in a volume increase in the ILLIAC IV. It is estimated that an additional 15 feet will be required per quadrant row. As a result of the volumetric increase a slowdown in the clock system must occur. It is estimated the clock will slow down to about 60 nsec.

Plans for Next Quarter

The major efforts during the next quarter will be to complete the PE design using DIL's to award a contract for the PE memories and to provide detailed information of the above problems on cost, schedule and performance. There have been no significant personnel changes during the last quarter. Current funding is adequate to carry the Project through mid-March.

1. HARDWARE

1.1 Logic Debugging and Diagnostics

1.1.1 Simulation and Debugging of PE Logic

The logic simulation and debugging of the Processing Element design have been in progress during this quarter. Two programs were written, in this period, to assist in the procedure of debugging - one for counting the logic levels and the other for identifying Boolean equations of major PE signals.

1.1.1.1 Improvement of the Logic Simulator

A few changes were made in the logic simulator program to reduce its running time. The simulator, as well as the Assembled Code Translator, received some changes caused by the altered, sign bit convention.

1.1.1.2 Level Counting Program

A modified version of the logic simulation program, which counts the logic levels associated with the control logic of the PE, was written for the purpose of estimating the time delay and debugging the control logic design. The procedure call statements associated with the control logic were selected out of the logic simulator body, and the parameter type was changed from Boolean variable to integer variable. The new procedures for calculating the logical levels were for all package types, and some additional coding was done to complete the level counting program. It was developed according to a request from the Burroughs' PE designer.

1.1.1.3 Equation Generator Programs

The DEG (DIL Equation Generator) writes an equation for dual-in-line packages in the PE. For the latch, DIL type 8, only the signal names are listed. The equation is written using the signal names associated with the package. Input to the program is the wire list which is scanned to separate DIL from MSI packages. The output of this program serves as input to the PE equation generator and can be used as a debugging aid.

The PEG (PE Equation Generator) develops an equation for P-signals in the PE control logic in terms of primitive inputs. A primitive input is a signal external to the PE or it is the output of a latch or an MSI package. The program is an extension of DEG in that it steps through the list of DIL equations generated in DEG and replaces any non-primitive input signals with the equation for the signal in terms of primitive inputs. The output of this program can be used as a debugging aid for the control logic of the PE.

1.1.1.4 Debugging

The simulator program was executed on test inputs from three sources:

(a) card decks of single cases manually defined,

(b) PE Exercisor test routines (from Burroughs) translated by the Assembled Code Translator, and

(c) automatic extraction of microsequences from POSFILE (PE instruction timing sheets).

Mechanization of basic arithmetic instructions, including logical addition and subtraction, fixed point multiplication, and exponent arithmetic, was debugged with card decks. PEX routines for Boolean and shift instructions, mode register instructions, eight-bit mode arithmetic, and some other instructions have also been tested on the simulator. The POSFILE approach was used to debug normalization and floating-point arithmetic instructions. Approximately 50 errors were found by the simulator by the end of this quarter. The major instructions to be debugged in the next quarter include division and some arithmetic instructions with variants.

1.1.2 CU Logic Simulator System

1.1.2.1 Overview

Specifications have been drawn up for two simulator systems to assist in the logical design and debugging of the Control Unit. These are the CU Card Logic Simulator System and the CU Section Logic Simulator System. Although the construction of these simulators will be similar to

that of the PE Simulator described in the previous QPR, several improvements have been included which should yield much greater simulation efficiency. Among these are a capability for parallel simulation of many test cases at the same time and a new simulation control language to permit simple specification of complex simulations.

1.1.2.2 CU Card Logic Simulator System

The purpose of the CU Card Logic Simulator System is to assist in debugging CU printed circuit card designs and to calculate expected responses for various test cases. In addition, it will be used to develop card simulation procedures for inclusion into the CU Section Logic Simulator System. The card simulator system will also incorporate a capability for the modification of the CU Card Logic Simulator to simulate the effect of logic failures. This will help in developing input patterns to be used for production tests and off-line diagnostics of the CU cards.

The CU Card Logic Simulator System will consist of four major groups of programs: the simulator generator group, the data preparation group, the simulator itself, and the results display group. The simulator generator group of programs are similar to those for the PE simulator. They consist of programs which reduce the CU card to a directed graph, make preliminary level assignments, detect loops, perform final level assignments, and then generate the actual simulator body. Two improvements for increased simulation efficiency have been added: logical simulation of the dual-inline packages is performed by in-line coding rather than by time-consuming procedure calls, and loop members are simulated along an optimal (Hamiltonian) path (if one exists) rather than in random order as in the PE simulator.

Much of the CU card logic simulator is similar to the PE logic simulator. However, a new, more flexible simulator input/output system has been designed. Also, the simulator can operate in modes which are not physically realizable but nevertheless are of potential use in logic debugging (e.g., registers can be prevented from changing state and can be preset to any desired contents). Further, the simulator has been designed with two additional goals in mind: first, to provide parallel simulation of up to 47 independent test cases by taking advantage of the fact that the

B5500 performs Boolean operations on full (47-bit) data words, and second, to permit simulation of logic failures by simple modifications to the simulator body.

A new simulation test control language, TESLA, has been developed. This test language permits simple description of desired simulations. The language is very general; its later use with the CU Section Logic Simulator System is anticipated. One unusual feature of TESLA is the incorporation of language elements for simultaneous generation and control of many simulations, thus complementing the parallel simulation capability of the simulator itself.

Since testing and fault diagnosis for the actual CU cards will be performed on the PEX (with a CU card tester adapter), it is desirable to be able to simulate these tests. This will be done by providing a method for translating PEX programs into simulator input.

On December 2nd, a meeting was held with Burroughs designers at Paoli. The proposed card simulator system was reviewed, and several helpful changes and improvements (particularly to TESLA) were suggested. Since that meeting, the design of the CU Card Logic Simulator System has essentially been completed, and coding of the necessary routines has begun. CU cards will be simulated as soon as the necessary netlists become available, with precedence being given to cards from those sections of the CU which will be completed earliest.

1.1.2.3 CU Section Logic Simulator System

Preliminary specifications for the CU Section Logic Simulator System have been developed. It will simulate the logical performance of one entire CU section (FINST, ADVAST, ILA, MSU or TMU). The CU section simulator will incorporate the bodies of the card simulators as procedures. Actual parameters to these procedures will be variables corresponding to the backplane signals.

Each section will also have a functional simulator developed for it. This functional simulator will be identical to the (correctly functioning) CU section logic in external performance, but will hopefully be far simpler in internal construction and thus much more efficient. It will

be constructed, insofar as possible, from narrative descriptions of the desired section performance rather than from actual logic diagrams. Thus, it should be somewhat insulated from simple design errors and should provide a valuable performance standard and source of expected responses.

Since writing a correctly operating, functional simulator exhibiting the same full range of dynamic performance as the actual logic will be no simple task, a bootstrapping technique will be used to permit continuous interdependent upgrading and improvement of the five functional simulators until the desired level of performance is achieved.

An important advantage which will accrue from the development of the functional simulators will be the ability to use actual assembled ILLIAC IV code to test any (simulated) section of the CU. Functional simulation of the other sections will provide the necessary translation of input code into the actual commands and data which are sent to the section under test.

Because a correctly functioning card simulator system is a necessary preliminary to the development of the section simulators, only initial efforts at defining some of the more global aspects of the CU Section Logic Simulator System have thus far been possible. However, substantial portions of the section simulator system design should be completed during the next quarter as designer manpower becomes available and as operating experience with the CU Card Logic Simulator System provides valuable feedback to the design process.

1.1.3 PE Diagnostics Generation

1.1.3.1 Path Tests

The debugging of the expected response calculator for the path test was continued, and the corrections of several errors were made in this period. The path tests will be finalized after the debugging of the PE logic.

1.1.3.2 Combinational Tests

Due to the recent shift from the implementation of the PE in MSI packages to the DIL approach, the test patterns for combinational logic will be reworked in the next periods.

1.2 Design Automation

The major effort of the design automation group, during this quarter, was giving substantial programming support to Burroughs, Paoli, Pennsylvania. The Mohawk data set is continuing to be used for design automation production jobs.

A single board Post-Processor was written and debugged at the University. This Post-Processor, which has new and enlarged specifications, is capable of examining each wire on the PC Board. Work has also begun on a Post-Processor capable of handling the entire CU with inter-board wires.

2. SOFTWARE

2.1 Operating System

The operating system provides several services each of which is implemented by one or more program modules within the system. The user interface to the operating system is through the job parser which can be driven by any available input media such as tapes, card decks, user operated consoles, or other running B6500 programs.

The collection of programs and some of the data needed on the ILLIAC IV is performed within the B6500 by a program collector.

The scheduling of ILLIAC IV time is performed by three system modules: the disk file allocator, the data pre/post processor, and the job execution monitor. These three programs queue up requests for ILLIAC IV use and assign Disk IV space to them, move any files that are needed onto the disk before a run, save any files by moving them off the disk after a run, schedule the use of BIOM space, and allocate particular ILLIAC IV quadrants to the individual jobs.

There are two main ILLIAC IV resident modules, the loader and OS4. The loader loads program files on Disk IV into the array memory. OS4 provides the standard monitor functions and the I/O execution coordination routines needed by all users.

The running ILLIAC IV job requires at least two intercommunicating programs -- the ILLIAC IV resident program and a B6500 resident job partner that coordinates B6500 actions with the ILLIAC IV program and provides all I/O support. Each job must have at least one job partner for all of its quadrants and each job may have as many job partners as it has separate quadrant code streams. A user may choose not to use the system-supplied job partner with the execution of his ILLIAC IV program and may write his own job partner.

The building of I/O descriptors, the initial recognition of interrupts, and the actual issuing of I/O commands to the hardware are functions of a set of routines in the B6500 MCP which are collectively known as the hardware supervisor. Since a user may write his own job partner,

the hardware supervisor also checks all of the I/O requests made by a job partner for validity before passing them onto the hardware.

Most of the modules named above are separately running B6500 ALGOL programs. They communicate with each other through the use of the in-core file facility provided by the B6500 MCP. The hardware supervisor routines (called "MCP intrinsics") are coded in ESPOL to facilitate both the issuing of I/O descriptors and the passing of interrupts to the correct job partner. The ILLIAC IV loader and OS4 are coded in the ILLIAC IV assembly language.

A preliminary version of OS4, the ILLIAC IV resident monitor, is being written and debugged. It soon will be published as a document.

2.2 Translator Writing System

2.2.1 Syntax Preprocessor (BNF → FPL)

During this quarter, the Floyd production and parser pseudo-order generator of the preprocessor were completely rewritten to implement improvements in the algorithm and effect the removal of empty productions and singly defined nonterminals by back-substitution.

The first stage of documentation, which describes the algorithm used in developing the syntax preprocessor, has been completed [1].

The first version of the syntax of Twinkle, a syntax specification language, was punched and partially debugged. During the next quarter, work on Twinkle will be stepped up since the syntax of Twinkle, with comments, will be the next stage of the documentation on the use of the syntax preprocessing portion of the TWS.

2.2.2 Parser (FPL)

The implementation of the improved BNF to FPL conversion algorithm necessitated changes in the parser interpreter. These changes were made and debugged which resulted in faster execution speeds.

The program CONVERT/TWS, which takes the table of parser pseudo-instructions generated by SYNPRØF/TWS and converts them to ALGOL, was written and debugged. Prior to the latest changes it achieved parsing times

equal to the scanning times. Early in the next quarter, it will be updated to convert the most recent output from SYNPRØF/TWS. A program is being contemplated which will automatically update the program CØNVERT/TWS to solve the problem of the long time lag between the improvements in the parsing algorithm and the correction of CØNVERT/TWS to handle them.

2.2.3 Semantics

During this quarter, the TWS-built ISL translator was completed and debugged. Consequently, the TWS system now has two complete ISL translators: (a) the "brute-force" ISL translator, and (b) the TWS-built ISL translator.

The advantages of (b) over (a) are basically two. First, the TWS-built ISL translator, having been implemented with the TWS system, is easily modifiable by anyone familiar with such a system. This is a very important feature, as it facilitates future extensions of ISL. Secondly, (a) implements a more sophisticated version of OSL than the basic ISL implemented in (b).

The main disadvantage of (b) over (a) is that (b) is considerably slower than (a), as could be expected, since (a) was very carefully optimized by hand. As the TWS system improves, however, it is hoped that the speed of (b) will approach that of (a), and (b) will become the only ISL translator. Tests are now being performed on the speed of the two translators.

Work is also continuing on the documentation of the language and by the end of next quarter a complete ISL manual should be available.

2.3 Compilers and Translators

2.3.1 Tranquil

During this quarter, work continued on the Pass 2 semantics for the Tranquil compiler. Progress on the Tranquil compiler has been delayed by the partial unavailability of the B5500. Documentation of Tranquil specifications and implementation techniques is now available in the form of three MS theses [2,3,4].

Routines which handle all storage allocation and compile time memory maintenance are fully coded and debugged. Some effort at improving the efficiency of the ALGOL coding of these routines has been initiated.

The complex substructure necessary for the analysis of assignment statements has been coded and debugged. Algorithms for code generation for assignment statements have been designed and are partially coded.

The code generation for SIM and SEQ statements has been completed. The algorithms for set storage allocation and usage have been revised to facilitate the usage of a set both as a mode bit pattern and as explicit numbers. The coding for the nondynamic cases of these revised schemes also has been completed. The algorithms are easily extendable for implementation of the dynamic costs.

Several straight forward but time-consuming routines of a general nature have been written. Among these are run-time arithmetic conversion routines for all combinations of conversions involving 32 and 64-bit floating point numbers, 48-bit signed integers, and both compile and run-time translations between set storage forms.

2.3.2 Glypnir

Features which allow routing and indexing have been implemented and debugged during this quarter. Several new debugging features have been added and a number of changes were made in the syntax to improve error detection and recover.

In addition, several changes were made in the semantics in an attempt to increase the speed and lower the core storage requirement, but these met with little success. B5500 mix time has been improved significantly, but the size and diversity of the compiler make it difficult to improve the computation time except by completely reorganizing the compiler.

The code which will implement subroutines has been completed but not debugged. The necessary changes which will allow the compiler to accept this code have been completed. However, due to a lack of PRT locations, a sizable delay may be necessary before this code can be inserted into the compiler.

The current compiler consists of approximately 8600 lines of ALGOL, has a core estimate of 15.8K words, and runs at approximately 350 cards per minute on the B5500.

2.3.3 Squash

Work has continued on the debugging aid for ALGOL, with expected completion in January 1969. An initial translator has been formed from the complete syntax and partial semantics. This is being debugged. The remaining semantic actions are being coded for incorporation into the translator early in January.

2.4 Scheduler and Simulators

2.4.1 Scheduler

The Scheduler for ILLIAC IV has been flow charted and a test version for the B5500 is being written. An up-to-date reference manual on the assembler was written and will be published by Burroughs. A special version of the assembler for use on the B5500-16PE ILLIAC IV at Paoli was written.

2.4.2 Simulators

During the past quarter, the new simulator was virtually completed. This simulator includes a simple loader that utilizes the loader pseudo-operations emitted by the assembler. Other features of the simulator are:

1. The simulator is half-load proof--the simulator may be restarted from a previous break-point if the execution is terminated.
2. Disk I/O facilities are included.
3. There will be a format and list facility to generate line printer output for debugging purposes.
4. There is an octal dump of all registers and memory.
5. A simulation scheduler has been written and demonstrated. The scheduler queues simulation requests and automatically initiates the simulator on a new job when it completes a previous job.

6. The simulator will multi-process well with other programs. A simulation, an ALGOL compile, and R/C (R/C is a multi-user text editor) have been successfully multi-processed.

The simulation ratio between B5500 execution time and simulated quadrant ILLIAC IV execution time is slightly less than 10^6, which is somewhat better than was expected on the 7094.

2.5 Control Data ALGOL

A project to consider using the Control Data 6600 as the control computer for ILLIAC IV has been initiated. A study of the conversion of Burroughs' Extended ALGOL to Control Data ALGOL is now being made. An immediate goal is to have the assembler and simulator running on the Control Data machine.

Control Data ALGOL is essentially ALGOL 60 with the ACM proposed ALGOL 60 input-output conventions. Burroughs' ALGOL is an extension of ALGOL 60 with such constructs as partial word operators, imbedded assignments, bit-by-bit logical operations, and much exotic input-output. It also has such gross syntactic structures as WHILE statements, DO...UNTIL statements, and CASE statements.

A preprocessor for Burroughs' ALGOL is being written. This preprocessor is processing the gross syntactic differences, the DEFINE and FILL statements, and many minor character set problems. A simple parser to generate subroutine calls for imbedded assignments and partial word operators will be added. Presently, work is being done on an interpretive method of simulating the Burroughs STREAM PROCEDURES. Eventually, the output of the preprocessor must be modified by hand to implement input-output calls and to change certain other constructs, which are not readily machine convertible, into CDC ALGOL.

2.6 CAT

2.6.1 Mesh Storage

For non-core-contained partial differential equation problems, the mesh of data points is stored on ILLIAC IV disk rather than in fast memory. Blocks of the mesh are brought into fast memory one at a time for processing. After the processing of a block (kernel calculations) is completed, processing of the next block in sequence is started. If the blocks do not span either dimension of the rectangular mesh, then two sweeping sequences are possible: sweeping by rows of blocks and sweeping by columns of blocks. In sweeping by rows, all of the blocks in a row i are read from disk in sequence. Then the blocks in row i + 1 are read in the same sequence. This process continues until all blocks in n rows are read. Sweeping by columns is similar.

Some problems may require successive sweeps of the mesh in alternating directions. Storing the blocks on disk in such a way that latency is minimized is a problem which is further complicated by the necessity for interblock communication. For the kernel to update a block, edges from the four neighboring blocks are required.

A scheme has been formulated which allows sweeping in any sequence of directions (by rows or by columns). Blocks are read from disk; kernel calculations are performed; and the updated blocks are written on disk. The scheme will accommodate a wide range of rectangular mesh sizes. Many particular schemes have been found over a sample of mesh sizes and with a latency ≤ 0.12 (12% of total problem time). The report of the investigation will be completed in January of 1969.

2.6.2 Subset of CAT

A model is being developed for optimizing code which is amenable to solution by linear programming. The model allows for reordering of statements from the original code and for many forms of non-obvious data manipulation to minimize the time the processor(s) waits for I/O.

This method is completely general in that it assumes no particular machine configuration. The necessary information, such as computation time for the various operations and secondary memory cycle time, are parameters. The number of (binary integer) variables resulting from any non-trivial code, unfortunately, is large ($\geq 10^3$). Present efforts are involved in attempting to reduce this number.

3. APPLICATIONS

3.1 Mathematical Applications

3.1.1 Partial Differential Equations

In this quarter, the Tranquil version of an idealized particle-in-cell code for a plasma was completely debugged for syntax errors. The code is contained in ILLIAC IV Document No. 207 [5] which was written this quarter. Document 207 gives a description of an idealized particle-in-cell (PIC) code for a plasma.

The checkerboard storages scheme for the PIC codes was investigated at Los Alamos Scientific Laboratory. For 256 PE's using a 16x16 checkerboard on meshes of 20x100 or 30x100, the efficiency, due to distribution of particles, ranges from 33% to 43% for 30 time steps. If, instead of using 256 PE's, only 64 PE's are used, the efficiency ranges from 80% to 90%.

A preliminary investigation into alternate storage schemes for PIC type problems was completed this quarter. Written this quarter was another code and document concerning the modified successive overrelaxation (MSOR) method for solving both Laplace's and Poisson's difference equations [6]. The MSOR is much more efficient for small meshes on a parallel computer.

3.1.2 Generalized Ordinary Differential Equation Solver

The general aim of this work is to produce an ALGOL program which will be run on the B6500 and which will take some simple representation of a large system of ordinary differential equations as input and produce as output the ILLIAC IV machine code to perform the integration. Before generating the code, this program must decide the best method and storage allocation for the given problem. Since the program must also scan the input and do some compiling, one of the Translater Writer Systems on the B5500 is being used to build a scanner written in ALGOL. Work is proceeding on the algorithms necessary to perform this task and in studying the types of methods available. In the main, however, the methods will already exist as

skeleton codes which will be completed to form the desired machine language program.

3.1.3 Multidimensional Compressible Hydrodynamics

New methods for the solution of multidimensional, compressible, hydrodynamic flow problems are being investigated. In particular, methods having greater parallelism than those currently in use are desired. Much of this quarter was devoted to gaining experience with existing Eulerian methods and various velocity and density weighting schemes including OIL, donor, and Rich without the use of artificial viscosity. Also, preliminary results on a new continuous rezone technique have been obtained.

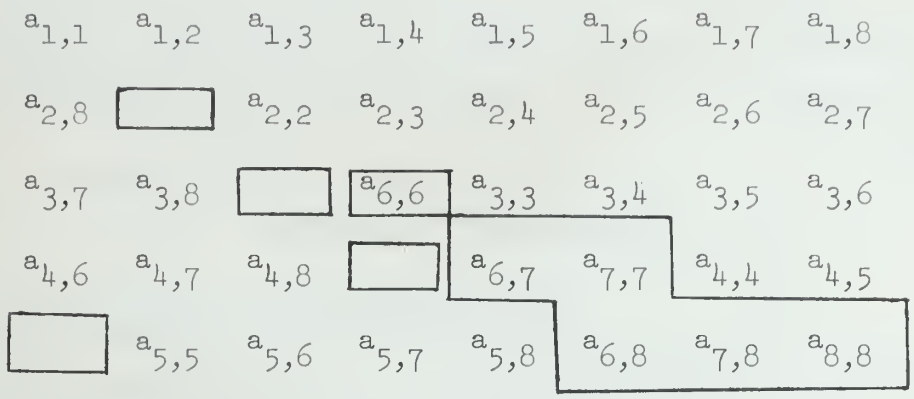
The above described techniques have been applied to test problems on the B5500. The test problems were selected from a report by Hicks of AFWL in which he describes several one-dimensional shock problems having analytic solutions [7]. The results obtained from "the shock tube problem" (V of the Hicks' report) are satisfactory. Greater difficulty has been encountered with the collision of two shocks (VI of the Hicks' report); although, the results obtained are comparable to those given by Lax-Wendroff.

3.1.4 Eigenvalues

3.1.4.1 Matrix Storage Methods

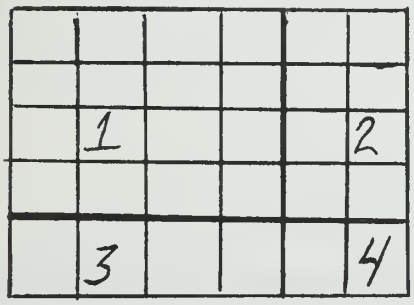
3.1.4.1.1 Matrix Storage for Jacobi's Method

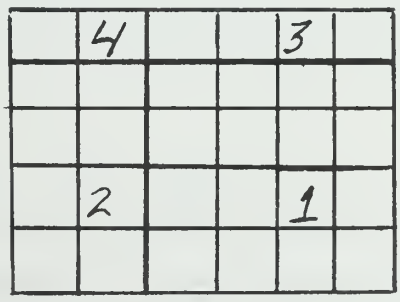
A new method of matrix storage has been developed for Jacobi's method of finding eigenvalues for a matrix [8]. The storage method used is a variation of the triangular skew method. To illustrate this method, look at an 8x8 matrix stored in this manner:



In this storage method, each pair of successive diagonal elements is separated by one PE. The diagonal element $a_{\frac{n}{2}+1,\ \frac{n}{2}+1}$, in this case, $a_{5,5}$, is an exception in that it is separated from the previous diagonal element by 2 PE's. The elements beginning with $a_{\frac{n}{2}+2}$, $a_{\frac{n}{2}+2}$ form a wedge in the lower part of the matrix, as illustrated in the diagram above.

Assume that the origin of the matrix storage is the storage location of the element $a_{1,1}$ in the beginning. Call this location (0,0). After each shuffling of the second row and column, the origin is moved two locations to the right and one down. After one shuffling, the origin would be at (1,2) and after two shufflings, the origin would be at (2,4). The origin is again at (0,0) after $\frac{n}{2}$ shufflings. The effect of this on the updated matrix is to make the last two columns the first two columns of the new matrix, and make the last row the first row:





In this way, the elements of the updated matrix retain the same relationships as in the previous matrix.

Since the matrix has to be symmetric for Jacobi's method, this storage scheme allows all elements to be stored in an economical way. More precisely, for an n x n matrix, only $\frac{n}{2}+1$ rows of memory are required.

3.1.4.1.2 Matrix Storage for QR - Algorithm

Because the ordinary QR-algorithm is extremely inefficient for ILLIAC IV, a modification which does many origin shifts (instead of the customary two) in a single iteration, is being developed. The effect of the modification is to fill the matrix and allow a full Householder reduction at each step, making the method much more suitable for ILLIAC IV. Numerical experiments indicate that, although it is slower than ordinary QR by a factor of two, the accuracy of the eigenvalues from the two methods is equivalent.

3.1.4.2 Extended ALGOL Codes and Programs

Several codes and programs were written in Extended ALGOL. Codes have been written during this quarter, for Bessel function of first kind $J_p(x)$ and Arctan (x). A program for finding the eignvalues and the eigenvectors for a symmetric matrix by using Jacobi's method was written in Extended ALGOL. Also written was a program to evaluate a matrix inversion by partitioning.

3.1.5 Polynomial Root Finding

During this quarter, an algorithm for determining whether a given square has a root in it was developed. There is a lemma in complex analysis which states:

$$S_n = \frac{1}{2\pi i} \int_c z^n \frac{f'(z)}{f(z)} \, dz = \sum_{c=1}^{v} z^n \; ;$$

where z_i (i = 1,2, ..., v) are all the zeroes of f(z) which lie in the interior of the closed contour C. If N = 0, then S_o either has a value of zero or some positive integer which corresponds to the number of roots of the polynomial in the contour. S_o may then be approximated by the

trapezoidal rule. Thus, the algorithm consists of subdividing a given square region into 64 smaller squares, integrating on each of the squares, and repeating the process on those squares yielding a positive integration.

A Tranquil code for polynomial root finding is being debugged. When this is completed, a document will be written.

3.1.6 Special Functions Subroutine Library

Investigation was begun on writing subroutines in double precision or 128-bit mode. Meanwhile during this quarter, work continued on the development of a special functions subroutine library. A new 64-bit subroutine has been written for arctangent of a quotient of two numbers. The signs of the divisor and dividend determine in which quadrant the result will be. It was decided that the dividend should be in the A register and the divisor in the B register.

Also, codes have been written in the 32-bit mode for the exponential, sine, cosine, natural logarithm, arctangent, and square root functions. All codes previously mentioned in Quarterly Progress Reports have been written in ILLIAC IV assembly language. They have been successfully assembled and are waiting to be simulated. Documentation concerning all 32-bit and all 64-bit special function subroutines is being prepared and should be completed shortly.

3.1.7 Random Number Generators

In this quarter, work began on the investigation of different random number generators for ILLIAC IV. The most popular computing scheme to generate random numbers uses the congruential relation

$$x_{i+1} \equiv ax_i + c \pmod{m},$$

in which x_o is a positive integer, called the starting value; a is integer, called the multiplier; c is another integer; and m is a fourth integer, called the modulus, which is positive and greater than the other three in magnitude. When c = 0, the method is called multiplicative congruential; otherwise, it is called mixed congruential.

The advantages of the two methods will be discussed in a forthcoming document. The main concern of the document will center on the possible choices of the four parameters which will enable the random numbers to satisfy the statistical tests required of them.

3.1.8 Significant Digit Arithmetic

The work to implement significant digit arithmetic (SDA) on ILLIAC IV was begun this quarter. From the viewpoint of error analysis, the conventional normalized floating point arithmetic has several facets, and confusion often results from the failure to distinguish how many significant digits the evaluated result has. However, SDA operations, which are performed using unnormalized floating point arithmetic, are intended to facilitate the identification of significant digits in the result. In other words, the object of SDA is to keep track of as many significant digits throughout the whole arithmetic operation as there are given in the initial values.

During this quarter, some basic ideas for implementing Metropolis-type SDA on ILLIAC IV were proposed, and several versions for operating SDA have been tried. These results, including the mathematical basis and the assembly language codes for multiplication and division, are summarized in a document which will be printed soon.

3.1.9 Long Codes

In this quarter, the stability behavior of the autonomous system $\dot{x}(t) = Ax(t)$, where A is a constant matrix, was studied [9]. Also studied was the linear Hamiltonian system which is derived from the above general system. The algebraic condition on A is $A = J.S$ where $J = \begin{array}{c|c} 0 & I \\ \hline -I & 0 \end{array}$, and S is the time independent symmetric matrix $(H_{x_v x_{v+n}})$, $v = 1,2, \ldots, m$, where $m = \frac{n}{2}$.

During this quarter. an elementary algorithm was suggested for finding the eigenvalues of the constant matrix A. A fundamental matrix X

is to be built up from n-linearly independent observation vectors at a given time t, x_1, x_2, ..., x_n. The matrix A is formed by the multiplication $\dot{X}X^{-1}$. Once the matrix A is formed, the QR-algorithm would be used to determine the eigenvalues, and consequently, the stability behavior of the system.

Furthermore, the stability behavior of the general linear systems $\dot{x}(t) = A(t)\, x(t)$ was studied in detail. The elements of A(t) are assumed to be continuous in t and periodic with the same period, and the nature of the solutions of such systems described by the Floquet theory [10,11,12].

The stability criteria of the associated linear Hamiltonian systems,

$$\dot{x}_v = H_{x_{v+n}}, \quad \dot{x}_{v+n} = -H_{x_v} \qquad v = 1,2, \ldots, m$$

where $\dot{x}(t) = J.S(t)\ x(t)$, were investigated by using the novel approach of J. Moser [13]. In his paper, Moser showed that the eigenvalues of the monodromy matrix of a special fundamental matrix of the system

$$\dot{x}(t) = J.S(t)\ x(t)$$

constitute the invariants of the above system. A stability criterion could be easily established in the case of distinct eigenvalues; however, in the degenerate cases, one needs to refine the stability criterion. This may be done by establishing an additional set of invariants of the system, and the conditions of stability can be expressed in terms of the additional set of invariants.

3.2 Linear Programming

3.2.1 Introduction

Much of the work involved in the development of the LP system is of a continuing nature, and topics discussed in the last QPR have been subject to re-examination and revision from time to time. This condition will continue to occur. During this quarter considerable progress was made by the LP group. Formal definitions of the LPS algorithm and the

associated data manipulations were almost completed. Some of the LP procedures were coded in ILLIAC IV assembly language and other coding is in progress. The progress to date is summarized below.

3.2.2 Data Input

The initial steps required to define and develop a formal language for use in matrix generation were taken. The requirements for a meaningful matrix-generation system were examined, and there will be a major effort in this area in the near future.

3.2.3 Preprocessing of Data

The procedures necessary to pack and store input data have been defined. The non-zero coefficients will be assigned to specific PE's and will be tagged accordingly for subsequent disk storage. The preprocessing will be done on the B6500.

3.2.4 Storage

The data storage procedure has been well defined and is as follows. Since individual columns of the coefficient (A) matrix are needed for d_j (reduced price) computation, for updating with previously computed η (product form inverse) vectors, and for subsequent entry into the basis, each column should be stored so that its non-zero coefficients are distributed across PE's. This is attained by first assigning rows to adjacent PE's, starting with the most dense, in such a manner as to evenly distribute the non-zero elements of A among PE's. This is subject to the constraint that elements of a given row can be assigned to only one PE. The coefficients of the rows assigned to each PE are then sorted by columns within the given PE. This storage scheme assures that operations done on columns will be done in parallel, since the elements of a given column will be stored in several PE's.

3.2.5 The LPS Algorithm

Progress in connection with the algorithm has been significant. The procedures for d_j computation, multiple pricing, pivot selection,

and vector updates using the product of previously generated η-vectors have been coded in ILLIAC IV assembly language. The mathematical procedures involved in right-hand side ranging, parametric programming, and bounding of variables have been examined and flow charted. An intensive examination of inversion techniques for use in the needed REINVERT subroutine has been initiated, and it is continuing. The subject of tolerances on computed values has been approached, and will be more carefully examined.

3.3 Radar Processing Applications

Efforts during this period have been devoted to the Kalman Filter Code debugging, to understanding a new, simpler version of Kalman Filtering developed by Lincoln Labs, and to converting the code for the designation mode to 32-bit floating point operation. Work began on programming Fast Fourier Transform in 32-bit floating point for ILLIAC IV to be used in the discrimination mode and on investigating the problems in radar scheduling and the effects on using ILLIAC IV for the scheduling.

Progress in the debugging of the Kalman Filtering program in ILLIAC IV assembly language has been slow. There is presently no assembler simulator that can handle the 32-bit floating arithmetic which is used in the Kalman programs. The program has been assembled on a comparable assembler and is essentially ready for the execution simulator. To facilitate faster and easier debugging of the assembly language code, the Kalman Filer Program has been written in ALGOL to run on the B5500 computer. This program will supply data for checking out each section of the assembly language program. The ALGOL program is running, but it is not completely debugged. With the aid of data supplied by Lincoln Labs, this program should be working by early February. At that time, it is probable that the ILLIAC IV assembler and simulator for 32-bit mode arithmetic will be in operation. The ILLIAC IV assembly language of the Kalman Filter can then be debugged by using the simulator and data generated for the ALGOL version.

The Tranquil version of the Kalman Filter written for ILLIAC IV will not be debugged until the compiler is completed to the point that it can handle all the Tranquil statements used in the program. This will not

be accomplished for sometime; however, when complete, it will be a good test case for using higher level language as opposed to assembly language on ILLIAC IV for radar processing.

A new version of the Kalman Filter which uses only approximately 1/2 to 2/3 the time of the old version has been developed by Lincoln Labs. It does everything in spherical coordinates instead of the combination of spherical and cortesian coordinates. This new version will be programmed in ILLIAC IV assembly language using 32-bit floating point arithmetic.

The designation mode which was in fixed point arithmetic (described in ILLIAC IV Document 173 [14])is being reprogrammed by the new graduate student into 32-bit floating point arithmetic. This mode uses a .1 second and a 1 second filtering technique of a cloud in respect to cloud center to try to filter objects from shaft.

The Fast Fourier Transform (FFT) is being programmed for ILLIAC IV as a subroutine using 32-bit floating point arithmetic. It will be set up in the following two forms: 1. returns from the same object are spread across PE's, and the different object on which data is obtained goes down memory of each PE; and 2. the objects are spread across PE's, and data on each goes down memory. Many different forms of discrimination make use of FFT, and it will be needed for ILLIAC IV.

The radar scheduling problem and the efficiency of doing this portion of the radar processing task on ILLIAC IV is being investigated. Several portions of this problem seem to be sequential; therefore, it may be more efficient to handle a large part of the task on a standard computer which is controlling ILLIAC IV. However, the interplay of the two computers on this problem is being studied.

3.4 Seismic Signal Processing

During this quarter, a study of the programming requirements necessary to compile a seismic signal processing package was undertaken. This resulted in defining the specifications for approximately twenty-five programs. Several of these programs, such as those involving matrix manipulations, may be able to incorporate current programs from other study areas.

This investigation has indicated that to make the seismic programs most functional, they must be written so that they can be used in areas other than seismic processing with a minimum amount of modification. Some of the specialized seismic programs will have applications in other areas--such as programs for convolution, filter design, deconvolution, power spectrum analysis, and correlation analysis.

3.5 Weather

The finite difference portions of the General Circulation Model Benchmark provided by the National Center for Atmospheric Research have been coded in a combination of Tranquil and assembly language. These arithmetic subroutines must be connected by a machine language control program which will provide necessary parameters and which will make adjustments for unusual grid spacing caused by mountain blocking and stability criterion near the poles.

3.6 Graphics

3.6.1 Introduction

The graphic activity is divided into software and hardware sections. The software activity is concerned with developing algorithms and techniques to apply ILLIAC IV to graphic problems. A specific task is presented in the following software section. The hardware activity is concerned with the peripheral equipment for the ILLIAC IV computer. This equipment will be used to generate output which will be used by the programmer for debugging his program and providing graphic output in the form of contour maps and various graphs. Consideration is also being given to providing a graphic terminal to provide program status to the operators.

3.6.2 Software

A problem of particular, current interest in the area of computer graphics is the so-called hidden line problem. This problem is concerned with computer determination of the visible and invisible parts of three dimensional objects when viewed from an arbitrary point.

A simple, fast algorithm is essential because there are many applications in which it is desired to view a moving object of a CRT. However, the problem has proved to be unusually complex. The following two structures are mainly concerned:

1. three dimensional polyhedra consisting of irregular and opaque polygons, where only line segments with end-point pairs, one pair for each line, are assumed as the input data because of the construction of three dimensional objects from line drawing and

2. three dimensional objects including curved surfaces decomposed into a lot of triangles, where each decomposed triangle of the curved surfaces must be completely specified as the input data.

The algorithm for polygon generation is completely specified, and the list structure format is used for data structure of pictures.

To motivate the description of the algorithm of the hidden line problem, the factors which determine whether or not which parts of objects are visible can be considered. In the case of a single polyhedron, call it the local hidden line problem. For a compound structure of polyhedra, locally visible polygons of a polyhedron may become invisible or partially invisible by the polygons of another polyhedron. Call this the global hidden line. In either case, if the relationship between two polygons is determined, the hidden line problem may be solved. The relationship between two polygons is defined as follows:

(a) One is enclosing the other;

(b) One is involved by the other; or

(c) There is no intersection between them.

The program is being implemented on B5500. Additional documentation will be provided by a masters thesis being written in this area. The thesis should be available next quarter.

3.7 Statistical Packages

During this quarter, design was begun for a statistical system for ILLIAC IV. The orientation of the system will be toward techniques which are applicable to large-scale data bases and which are used repeatedly. This is in apposition to a multiplicity of small techniques which are oriented toward editing data or output or which are convenience programs with little computational content. It is intended that the computation aspects of the statistical system will rely heavily upon matrix operations generated by the matrix algebra group, when feasible.

The following functions have been chosen as a basic set:

(a) Data transformation,

(b) General Matrix operations,

(c) Correlation,

(d) Multiple Regression,

(e) Analysis of Variance
(by itself regarded currently as a very significant problem),

(f) Principal Axis Factor Analysis,

(g) Autocorrelations, and

(h) Frequency counting and Distribution Analysis.

It is expected that a limited number of functions may be added to this list in the future.

Under consideration, also are tools or languages for dealing with large scale data structures more complicated than the rectangular arrays customarily used for input to the above techniques. This system is still in the speculative stage.

3.8 ILLIAC IV Education

3.8.1 Introduction

The ILLIAC IV education for this quarter consisted of organizing and presenting two courses of instruction--one covering B5500 and

ALGOL material and the other covering material concerned with the ILLIAC IV machine, its languages, and its applications. In addition, the education group has been updating educational material for the use of the ILLIAC IV staff and persons interested in the ILLIAC IV. This quarter also saw an attempt to organize all programs written or being written for ILLIAC IV into a library, which is to be available to anyone interested.

3.8.2 B5500 and ALGOL

This course was offered to acquaint ILLIAC IV personnel with the B5500 and ALGOL. The topics covered in this course were: the basic elements of ALGOL and special B5500 Extended ALGOL features, descriptions and instructions for use of the peripheral units, a detailed description of I/O, and a discussion of the master control program of the B5500.

3.8.3 CS 491-D

This course was a graduate seminar course CS 491-D, which was offered to ILLIAC IV personnel. (It will be offered University-wide for the spring semester.) This course included a discussion of the hardware of ILLIAC IV, coverage of its assembly language and Tranquil, a discussion of the Assembler, and extensive coverage of ILLIAC IV applications.

REFERENCES

[1] Beals, Alan J. "The Generation of a Deterministic Parsing Algorithm." Master's Thesis, Report No. 304. Urbana, Illinois: Department of Computer Science, University of Illinois at Urbana-Champaign, 1969.

[2] Budnik, Paul P. "TRANQUIL Arithmetic." Master's Thesis, Report No. 296. Urbana, Illinois: Department of Computer Science, University of Illinois at Urbana-Champaign, 1969.

[3] Muraoka, Yoichi. "Storage Allocation Algorithms in the TRANQUIL Compiler." Master's Thesis, Report No. 297. Urbana, Illinois: Department of Computer Science, University of Illinois at Urbana-Champaign, 1969.

[4] Wilhelmson, Robert B. "Control Statement Syntax and Semantics of a Language for Parallel Processors." Master's Thesis, Report No. 298. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

[5] Rudsinski, L. E. "A Tranquil Version of an Idealized Particle-In-Cell Code for a Plasma." ILLIAC IV Document Number 207, (November 25, 1968).

[6] Rudsinski, L. E. "Tranquil Code for the Modified Successive Overrelaxation (MSOR) Method for Solving Laplace's Difference Equation." ILLIAC IV Document Number 208, (December 24, 1968).

[7] Hicks, Darrell. "Hydrocode Test Problems." AFWL-TR-67-127. New Mexico: Kirtland Air Force Base.

[8] Kuck, David. "ILLIAC IV Organization, Programming, and Storage Allocation." Lecture given before the Mathematics, Computation, and Computers Conference, University of Illinois at Urbana-Champaign, Urbana, Illinois, November, 1968.

[9] Malkin, I. G. "Theory of Stability of Motion." Translation. Moscow and Leningrad: State Publishing House of Technical-Theoretical Literature, 1952.

[10] Coddington, E. A., and Levinson, N. Theory of Ordinary Differential Equations. New York: McGraw Hill, 1955.

[11] Wintner, A. The Analytical Foundations of Celestial Mechancis. Princeton: Princeton University Press, 1941.

[12] Lefschetz, S. Differential Equations, Geometric Theory. New York: International Science Publishers, Incorp., 1957.

[13] Moser, J. "New Aspects in the Theory of Stability of Hamiltonian Systems," Communications on Pure and Applied Mathematics. XI, (1958), 81-114.

[14] Knapp, M. A., Ackins, G.N., and Thomas, John. "Application of ILLIAC IV to Urban Defense Radar Problem." ILLIAC IV Document Number 173, (February 21, 1968).

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

ILLIAC IV QUARTERLY PROGRESS REPORT
October, November, December 1968

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Oct. - Dec., 1968 - Progress Report of the ILLIAC IV Project

5. AUTHOR(S) *(First name, middle initial, last name)*

6. REPORT DATE

February 1, 1969

7a. TOTAL NO. OF PAGES

36

7b. NO. OF REFS

14

8a. CONTRACT OR GRANT NO.

46-26-15-305

b. PROJECT NO.

USAF 30(602)-4144

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

ILLIAC IV Document No.
DCS Report No.

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

RADC TR

10. DISTRIBUTION STATEMENT

Qualified requesters may obtain copies of this report from DCS.

11. SUPPLEMENTARY NOTES

NONE

12. SPONSORING MILITARY ACTIVITY

Rome Air Development Center
Griffis Air Force Base
Rome, New York 13440

13. ABSTRACT

See the Report Summary on Page 1 within the Report itself

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
PE Logic						
CU Logic Simulator System						
PE Diagnostics Generation						
Operating System						
Syntax Preprocessor						
Tranquil						
Glypnir						
Squash						
CDC ALGOL						
CAT						
Multidimensional Compressible Hydrodynamics						
Long Codes						
Linear Programming						
Radar Processing Applications						
Seismic Signal Processing						
Graphics						
Statistical Packages						